UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
09/842,270	04/24/2001	Michael J. Grier	2501	5601

7590 05/06/2004		EXAMINER
Law Offices of Albert S. Michalik, PLLC		YIGDALL, MICHAEL J
704-228th Avenue NE		

ART UNIT	PAPER NUMBER
2122	8

Law Offices of Albert S. Michalik, PLLC
704-228th Avenue NE
Suite 193
Sammamish, WA 98074

DATE MAILED: 05/06/2004

Please find below and/or attached an Office communication concerning this application or proceeding.

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) FROM
THE MAILING DATE OF THIS COMMUNICATION.
- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed
  after SIX (6) MONTHS from the mailing date of this communication.
- If the period for reply specified above is less than thirty (30) days, a reply within the statutory minimum of thirty (30) days will be considered timely.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133).
  Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any
  earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *24 April 2001*.

2a)☐ This action is **FINAL**.   2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is
closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1-49* is/are pending in the application.

    4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1-49* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☒ The specification is objected to by the Examiner.

10)☒ The drawing(s) filed on *24 April 2001* is/are: a)☐ accepted or b)☒ objected to by the Examiner.

    Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

    Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

    a)☐ All   b)☐ Some * c)☐ None of:

      1.☐ Certified copies of the priority documents have been received.

      2.☐ Certified copies of the priority documents have been received in Application No. _____.

      3.☐ Copies of the certified copies of the priority documents have been received in this National Stage
application from the International Bureau (PCT Rule 17.2(a)).

    * See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)

2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)

3) ☒ Information Disclosure Statement(s) (PTO-1449 or PTO/SB/08)
Paper No(s)/Mail Date *6,7*.

4) ☐ Interview Summary (PTO-413)
Paper No(s)/Mail Date. _____ .

5) ☐ Notice of Informal Patent Application (PTO-152)

6) ☐ Other: _____.

DETAILED ACTION

1.   Claims 1-49 are pending and have been examined. The priority date considered for the

application is 24 April 2000.

Specification

2.   The attempt to incorporate subject matter into this application by reference to the

copending application entitled "Configurations for Binding Software Assemblies to

Applications" is improper because the application serial number has not been provided (see page

16, lines 14-22).

Drawings

3.   The drawings are objected to as failing to comply with 37 CFR 1.84(p)(5) because they

do not include the following reference sign(s) mentioned in the description: 210 of FIG. 2A (see

page 17, lines 6-10). Note that the specification does not have a FIG. 2A, and FIG. 2 does not

show a dashed box as described for FIG. 2A.   A proposed drawing correction or corrected

drawings are required in reply to the Office action to avoid abandonment of the application. The

objection to the drawings will not be held in abeyance.

Double Patenting

4.   The nonstatutory double patenting rejection is based on a judicially created doctrine

grounded in public policy (a policy reflected in the statute) so as to prevent the unjustified or

improper timewise extension of the "right to exclude" granted by a patent and to prevent possible

harassment by multiple assignees. See *In re Goodman*, 11 F.3d 1046, 29 USPQ2d 2010 (Fed.

Cir. 1993); *In re Longi*, 759 F.2d 887, 225 USPQ 645 (Fed. Cir. 1985); *In re Van Ornum*, 686

F.2d 937, 214 USPQ 761 (CCPA 1982); *In re Vogel*, 422 F.2d 438, 164 USPQ 619 (CCPA

1970); and, *In re Thorington,* 418 F.2d 528, 163 USPQ 644 (CCPA 1969).

A timely filed terminal disclaimer in compliance with 37 CFR 1.321(c) may be used to

overcome an actual or provisional rejection based on a nonstatutory double patenting ground

provided the conflicting application or patent is shown to be commonly owned with this

application. See 37 CFR 1.130(b).

Effective January 1, 1994, a registered attorney or agent of record may sign a terminal

disclaimer. A terminal disclaimer signed by the assignee must fully comply with 37

CFR 3.73(b).

5.   Claims 1, 3-5, 7-22, 24-26, 31, 42, 43 and 45-49 are provisionally rejected under the

judicially created doctrine of obviousness-type double patenting as being unpatentable over

claims 1-4, 12, 16-22, 26 and 27 of copending Application No. 09/842,278. Although the

conflicting claims are not identical, they are not patentably distinct from each other because both

recite a method and a system for providing to executable code a particular version of an

assembly based on stored information.

For example, claim 1 of both applications recites receiving a request for an assembly

from executable code and determining a particular version of that assembly based on separate

information. Claim 16 of the present application further recites mapping an assembly to a

particular version, in the same way claim 2 of the conflicting application recites redirecting an

assembly to a particular version. Moreover, claim 42 of the present application is analogous to

claim 19 of the conflicting application, which includes the limitations of conflicting claims 17

and 18. Also for example, present claim 45 and conflicting claim 20 both recite persisting an

activation context, and present claim 18 and conflicting claim 21 both recite storing a manifest in

the same folder as the executable file.

This is a <u>provisional</u> obviousness-type double patenting rejection because the conflicting

claims have not in fact been patented.

Claim Rejections - 35 USC § 101

6.   35 U.S.C. 101 reads as follows:

> Whoever invents or discovers any new and useful process, machine, manufacture, or composition of matter, or any new and useful improvement thereof, may obtain a patent therefor, subject to the conditions and requirements of this title.

7.   Claims 32-41 are rejected under 35 U.S.C. 101 because the claimed invention is directed

to non-statutory subject matter.

Claims 32-41 are recited as data structures stored on computer-readable media. Where

certain types of descriptive material, such as arrangements or compilations of facts or data, are

merely stored so as to be read or outputted by a computer without creating any functional

interrelationship, either as part of the stored data or as part of the computing processes performed

by the computer, then such descriptive material alone does not impart functionality either to the

data as so structured, or to the computer. Such "descriptive material" is not a process, machine,

manufacture or composition of matter, and thus does not constitute statutory subject matter. See

MPEP § 2106(IV)(B)(1)(b).

Claim Rejections - 35 USC § 102

8.   The following is a quotation of the appropriate paragraphs of 35 U.S.C. 102 that form the

basis for the rejections under this section made in this Office action:

A person shall be entitled to a patent unless –

(e) the invention was described in (1) an application for patent, published under section 122(b), by another filed in the United States before the invention by the applicant for patent or (2) a patent granted on an application for patent by another filed in the United States before the invention by the applicant for patent, except that an international application filed under the treaty defined in section 351(a) shall have the effects for purposes of this subsection of an application filed in the United States only if the international application designated the United States and was published under Article 21(2) of such treaty in the English language.

9.   Claims 1-17, 19-22, 25-31, 42, 43, 45-47 and 49 are rejected under 35 U.S.C. 102(e) as

being anticipated by U.S. Pat. No. 5,974,470 to Hammond.

With respect to claim 1, Hammond discloses a computer-implemented method (see the

title and abstract), comprising:

(a) receiving a request from executable code to load an assembly, the request not

including assembly version data (see column 5, line 58 to column 6, line 12, which shows

receiving a request from an executable application to load a module or assembly);

(b) consulting information associated with the executable code to determine a particular

version of the assembly (see column 7, line 51 to column 8, line 5, which shows consulting

information associated with the application to determine a version of the DLL or assembly); and

(c) providing the particular version of the assembly for use by the executable code (see

column 5, lines 27-30, which shows providing the correct version of the DLL or assembly).

With respect to claim 2, Hammond further discloses the limitation wherein the request

corresponds to a request to load a privatized assembly (see column 5, lines 30-34, which shows

that the DLL or assembly may be located in the application's directory, i.e. the requested

assembly may be privatized).

With respect to claim 3, Hammond further discloses the limitation wherein the request corresponds to a request to load a shared assembly (see column 5, lines 52-57, which shows that the DLL or assembly may be shared by many applications).

With respect to claim 4, Hammond further discloses the limitation wherein the shared assembly is maintained in an assembly cache (see column 5, lines 52-57, which shows that the shared DLL or assembly is located in a designated directory, i.e. in an assembly cache).

With respect to claim 5, Hammond further discloses the limitation wherein consulting information associated with the executable code to determine a particular version of the assembly includes searching for a mapping from a version independent name provided by the executable code to a version specific assembly (see column 7, line 51 to column 8, line 5, which shows searching for a rule or mapping from the requested DLL name to a specific version of the DLL or assembly).

With respect to claim 6, Hammond further discloses the limitation wherein no mapping from the version independent name to a version specific assembly is present, and wherein providing the particular version of the assembly for use by the executable code comprises providing a default version (see column 6, lines 57-59, which shows providing the default module or assembly when no rules or mappings are present).

With respect to claim 7, Hammond further discloses the limitation wherein providing the particular version of the assembly comprises accessing a file corresponding to the assembly and loading the assembly into memory from the file (see column 5, lines 7-18, which shows that the

rules for providing the particular version of a DLL or assembly for loading into memory are

accessed from a file).

With respect to claim 8, Hammond further discloses the limitation wherein the

information associated with the executable code includes a mapping between a version

independent name provided by the executable code and a version specific file system path and

filename of the particular version of the assembly, and wherein providing the particular version

of the assembly comprises returning the path and filename to an assembly loading mechanism

(see column 5, line 58 to column 6, line 12, which shows returning the fully qualified path of the

particular version of the module or assembly to the loading routine).

With respect to claim 9, Hammond further discloses the limitation wherein the executable

code is stored as an application executable file in a folder, and wherein the version of the

assembly is stored as another file in the same folder (see column 5, lines 30-34, which shows that

the application is stored as an executable file in a directory and the DLL or assembly may be

located in the same directory).

With respect to claim 10, Hammond further discloses the limitation wherein the filename

corresponds to a file in an assembly cache (see column 6, lines 40-54, which shows that the fully

qualified path corresponds to a file in a shared directory, i.e. in an assembly cache).

With respect to claim 11, Hammond further discloses the limitation wherein the

information associated with the executable code is derived from application manifest (see

column 5, lines 7-18, which shows that the information is stored in a configuration file associated with the application, i.e. in an application manifest).

With respect to claim 12, Hammond further discloses the limitation wherein the information associated with the executable code is further derived from at least one assembly manifest (see column 5, lines 7-18, which shows that the information is stored in configuration files associated with the DLLs or assemblies, i.e. in assembly manifests).

With respect to claim 13, Hammond further discloses the limitation wherein the information associated with the executable code is constructed during a pre-execution initialization phase (see column 5, lines 19-25, which shows that the information may be constructed at any designated time, such as upon installation of the application, i.e. during an initialization phase prior to execution).

With respect to claim 14, Hammond further discloses the limitation wherein the information associated with the executable code is persisted into a non-volatile memory (see column 5, lines 7-18, which shows storing or persisting the information in a file, i.e. in non-volatile memory).

With respect to claim 15, Hammond further discloses a computer-readable medium having computer-executable instructions for performing the recited method (see column 5, lines 33-49, which shows applying software patches, i.e. computer-executable instructions, to an operating system inherently stored on a computer-readable medium).

With respect to claim 16, Hammond discloses a computer-implemented method (see the abstract), comprising:

(a) interpreting dependency information associated with executable code, the dependency information identifying at least one particular version of an assembly (see column 7, line 51 to column 8, line 5, which shows interpreting dependency information associated with an executable application to determine a version of a DLL or assembly); and

(b) associating with the executable code at least one mapping based on the dependency information, each mapping relating a version independent assembly name that the executable code may provide to a version specific assembly identified in the dependency information (see column 7, line 51 to column 8, line 5, which shows associating with the application a rule or mapping from the requested DLL name to a specific version of the DLL or assembly).

With respect to claim 17, Hammond further discloses the limitation wherein the dependency information is provided in an application manifest associated with the executable code (see column 5, lines 7-18, which shows that the information is provided in a configuration file associated with the application, i.e. in an application manifest).

With respect to claim 19, Hammond further discloses the limitation wherein at least one mapping maps a version independent name to an assembly stored in a common folder with an application executable file that corresponds to the executable code (see column 5, lines 30-34, which shows that the application is an executable file in a directory and the DLL or assembly may be commonly located in the same directory).

With respect to claim 20, Hammond further discloses the limitation wherein at least one

mapping maps a version independent name to a shared assembly in an assembly cache (see

column 6, lines 40-54, which shows mapping the requested name to a DLL or assembly located

in a shared directory, i.e. in an assembly cache).

With respect to claim 21, Hammond further discloses the limitation wherein the

dependency information provided by the executable code corresponds to an assembly having an

assembly manifest associated therewith, and further comprising, interpreting the assembly

manifest (see column 5, lines 7-18, which shows that the information is stored in configuration

files associated with the DLLs or assemblies, i.e. in assembly manifests).

With respect to claim 22, Hammond further discloses the limitation wherein the assembly

manifest specifies that a particular version of an assembly be replaced with another version of

that assembly (see column 8, lines 6-22, which shows replacing a version of a DLL or assembly

with another specified version).

With respect to claim 25, Hammond further discloses the limitation wherein the at least

one mapping is maintained in an activation context, and further comprising, persisting the

activation context (see column 8, lines 23-58, which shows alias mappings maintained in a

database, i.e. in an activation context; see also column 9, lines 25-40, which shows persisting the

activation context in the form of an alias file).

With respect to claim 26, Hammond further discloses the limitation wherein associating

with the executable code the at least one mapping comprises retrieving a persisted activation

context (see column 9, lines 25-32, which shows retrieving a persisted alias file or activation

context).

With respect to claim 27, Hammond further discloses the limitation wherein associating

with the executable code the at least one mapping comprises constructing a new activation

context (see column 9, lines 32-40, which shows constructing a new alias file or activation

context).

With respect to claim 28, Hammond further discloses the limitation wherein the new

activation context is constructed upon determining that an activation context does not exist (see

column 9, lines 32-40, which shows constructing the new alias file or activation context when

one does not exist).

With respect to claim 29, Hammond further discloses the limitation wherein the new

activation context is constructed upon determining that an existing activation may not be not

coherent with current policy (see column 9, lines 1-40, which shows determining whether a

loaded module, i.e. an existing activation, is the correct version, i.e. is coherent with current

policy, and then constructing a new alias file or activation context).

With respect to claim 30, Hammond further discloses running the executable code,

receiving a request from the executable code to load an assembly, the request including data

corresponding to a version independent name of the assembly and providing a particular version

of the assembly for use by the executable code based on a mapping therefor (see column 5, line

58 to column 6, line 12, which shows receiving a request from a running executable application

to load a module or assembly and providing the correct version; see also column 7, line 51 to

column 8, line 5, which shows determining the correct version of the DLL or assembly based on

rules or mappings).

With respect to claim 31, Hammond further discloses a computer-readable medium

having computer-executable instructions for performing the recited method (see column 5, lines

33-49, which shows applying software patches, i.e. computer-executable instructions, to an

operating system inherently stored on a computer-readable medium).

With respect to claim 42, Hammond discloses a system in a computing environment (see

the abstract), comprising:

(a) an initialization mechanism configured to interpret dependency data associated with

executable code, the dependency data corresponding to at least one assembly version on which

the executable code depends (see column 7, line 51 to column 8, line 5, which shows interpreting

dependency information associated with an executable application to determine a version of a

DLL or assembly needed by the application);

(b) an activation context, the activation context associated with the executable code and

constructed by the initialization mechanism based on the dependency data, the activation context

relating at least one version independent assembly identifier to a version specific assembly (see

column 8, lines 23-58, which shows an activation database or context associated with the

application that relates the requested DLL to a version-specific alias); and

(c) a version matching mechanism configured to access the activation context to relate a

version independent request from the executable code to a version specific assembly (see column

7, line 51 to column 8, line 5, which shows matching the requested DLL name with a specific
version of the DLL or assembly).

With respect to claim 43, Hammond further discloses the limitation wherein the
dependency data is included in executable code manifest (see column 5, lines 7-18, which shows
that the dependency data is stored in a configuration file associated with the application, i.e. in an
executable code manifest).

With respect to claim 45, Hammond further discloses the limitation wherein the
initialization mechanism persists the activation context (see column 9, lines 25-40, which shows
persisting the activation context in the form of an alias file).

With respect to claim 46, Hammond further discloses an assembly loading mechanism
configured to communicate with the executable code and the version matching mechanism to
load the version specific assembly upon a request by the executable code to load a requested
assembly, wherein the request does not include version specific data (see column 5, line 58 to
column 6, line 12, which shows loading the correct version of the module or assembly requested
by the executable application).

With respect to claim 47, Hammond further discloses the limitation wherein the assembly
loading mechanism loads the version specific assembly from an assembly cache (see column 5,
lines 52-57, which shows that the DLL or assembly is located in a shared directory, i.e. in an
assembly cache).

With respect to claim 49, Hammond further discloses a computer-readable medium having computer-executable modules configured to implement the recited system (see column 5, lines 33-49, which shows applying software patches, i.e. computer-executable modules, to an operating system inherently stored on a computer-readable medium).

10.   Claims 32-41 are rejected under 35 U.S.C. 102(e) as being anticipated by U.S. Pat. No. 6,185,734 to Saboff et al. (hereinafter Saboff).

With respect to claim 32, Saboff disclose a computer-readable medium having stored thereon a data structure (see the title and abstract, and FIGS. 5 and 6), comprising:

(a) a first set of data comprising a name of an assembly (see service 509 in FIG. 6, and column 9, lines 11-18);

(b) a second set of data comprising a version of the assembly (see version 513 in FIG. 6, and column 9, lines 53-55);

(c) a third set of data comprising at least one item of the assembly (see, for example, state 510 and type 511 in FIG. 6, and column 9, lines 19-39); and

(d) a fourth set of data comprising binding path data to each item in the third set of data (see path 507 in FIG. 6, and column 9, lines 5-7).

With respect to claim 33, Saboff further discloses the limitation wherein the binding path data comprises a location of a dynamic link library (see column 9, lines 53-55, which shows the location of a file, and lines 33-39, which shows that the file may correspond to a library, i.e. to a dynamic-link library).

With respect to claim 34, Saboff further discloses the limitation wherein the binding path data comprises an object class identifier (see identifier 506 in FIG. 6, and column 9, lines 1-4, which shows a unique identifier that may serve as an object class identifier).

With respect to claim 35, Saboff further discloses the limitation wherein the binding path data comprises a programmatic identifier (see identifier 506 in FIG. 6, and column 9, lines 1-4, which shows a unique identifier that may serve as a programmatic identifier).

With respect to claim 36, Saboff further discloses a fifth set of data comprising data corresponding to at least one dependency on an assembly (see dependencies 504 in FIG. 6, and column 8, lines 57-59).

With respect to claim 37, Saboff further discloses a fifth set of data comprising data corresponding to a Windows® class (see extensions 505 in FIG. 6, and column 8, lines 60-67, which shows data corresponding to the capabilities provided by a service, for example such as a Windows® class).

With respect to claim 38, Saboff further discloses a fifth set of data comprising data corresponding to a global name (see column 9, lines 11-18, which shows that the service name is an abstract or global name).

With respect to claim 39, Saboff discloses a computer-readable medium having stored thereon a data structure (see the title and abstract), comprising:

(a) a first set of data comprising a version independent name of an assembly (see service 509 in FIG. 6, and column 9, lines 11-18); and

(b) a second set of data comprising a filename path to a specific version of the assembly;

wherein the second set of data is associated with the first set of data such that a reference to the

version independent name in the first set of data is mapped to the specific version of the

assembly via the second set of data (see path 507 in FIG. 6, and column 9, lines 5-7).

With respect to claim 40, Saboff further discloses a third set of data comprising a version

independent object class name (see column 9, lines 11-18, which shows an abstract name that

may serve as an object class name), a fourth set of data comprising an assembly name

corresponding to a file that contains an object class that corresponds to the object class name in

the third set of data (see column 9, lines 5-7, which shows a path to a file), and a fifth set of data

comprising a version specific name that corresponds to the third set of data (see version 513 in

FIG. 6, and column 9, lines 53-55).

With respect to claim 41, Saboff discloses a computer-readable medium having stored

thereon a data structure (see the title and abstract), comprising:

(a) a first set of data comprising a version independent object class name (see service 509

in FIG. 6, and column 9, lines 11-18, which shows an abstract name that may serve as an object

class name);

(b) a second set of data comprising an assembly name corresponding to a file that

contains an object class that corresponds to the object class name in the first set of data (see path

507 in FIG. 6, and column 9, lines 5-7, which shows a path to a file); and

(c) a third set of data comprising a version specific name that corresponds to the first set

of data such that a reference to the version independent name in the first set of data is mapped to

the specific version of the object class (see version 513 in FIG. 6, and column 9, lines 53-55).

Claim Rejections - 35 USC § 103

11.   The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all

obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in
> section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are
> such that the subject matter as a whole would have been obvious at the time the invention was made to a person
> having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the
> manner in which the invention was made.

12.   Claims 18, 24 and 44 are rejected under 35 U.S.C. 103(a) as being unpatentable over

Hammond, as applied to claims 17, 16 and 42 above, respectively.

With respect to claim 18, although Hammond discloses a common directory storing both

the DLLs or assemblies and the executable file of the application (see column 5, lines 30-34),

Hammond is silent as to the location of the configuration file or application manifest (see column

5, lines 7-18). Accordingly, Hammond does not expressly disclose the limitation wherein the

application manifest is associated with the executable code by being stored in a common folder

with an application executable file that corresponds to the executable code.

However, is well known in the art that configuration files or manifests and other

application files may be stored in the same directory as the executable file. The advantage of

such an arrangement is that the operating system may find the files using the predetermined

search order (see Hammond, column 5, lines 30-34).

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to store the configuration file or application manifest taught by Hammond in a common folder along with the application executable file that corresponds to the executable code, so that the operating system may successfully find the file.

With respect to claim 24, although Hammond discloses interpreting dependency information (see column 7, line 51 to column 8, line 5) in response to a request from a running application to load a module or assembly (see column 5, line 58 to column 6, line 12), Hammond does not expressly disclose the limitation wherein the dependency information is interpreted in response to receiving a request to execute the executable code.

However, Hammond further discloses that the modules or assemblies are dynamic-link libraries, which are loaded and linked at run time, i.e. when the executable code is executed (see column 1, lines 23-32). It is well known in the art that DLLs or other assemblies may be required immediately upon execution of an application.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to perform the interpretation shown by Hammond in response to receiving a request to execute the executable code, because that request may require a particular version of a DLL or assembly.

With respect to claim 44, although Hammond discloses storing the dependency data in a configuration file (see column 5, lines 7-18), Hammond does not expressly disclose the limitation wherein the dependency data is included in an XML file.

However, it is well known in the art that XML is a flexible, standard markup language for structuring and organizing data.

Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to format the configuration file taught by Hammond as an XML file, for the purpose of structuring the dependency data using a standard language.

13.   Claims 23 and 48 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hammond, as applied to claims 21 and 42 above, respectively, in view of Saboff.

With respect to claim 23, although Hammond discloses dependency information for an executable application (see column 7, line 51 to column 8, line 5), Hammond does not expressly disclose the limitation wherein the assembly manifest specifies at least one particular version of another assembly on which the assembly having an assembly manifest is dependent.

However, Saboff discloses a registry structure that serves as an assembly manifest for managing versions of software components (see the title and abstract, and FIGS. 5 and 6), wherein the manifest specifies the assemblies upon which a first assembly is dependent (see column 7, line 61 to column 8, line 14). The dependency information specifies the files that are required by a library or assembly (see column 7, lines 61-64).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the assembly manifest of Hammond with the additional dependency information taught by Saboff, for the purpose of specifying the files required by a particular version of an assembly.

With respect to claim 48, although Hammond discloses dependency information for an executable application (see column 7, line 51 to column 8, line 5) and adding such information to an activation context (see column 8, lines 23-58), Hammond does not expressly disclose the limitation wherein the dependency data identifies an assembly that has assembly dependency data associated therewith, the assembly dependency data corresponding to at least one other assembly version on which the assembly depends.

However, Saboff discloses a registry structure that serves as an assembly manifest for managing versions of software components (see the title and abstract, and FIGS. 5 and 6), wherein the manifest specifies the assemblies upon which a first assembly is dependent (see column 7, line 61 to column 8, line 14). The dependency information specifies the files that are required by a library or assembly (see column 7, lines 61-64).

It would have been obvious to one of ordinary skill in the art at the time the invention was made to extend the assembly manifest of Hammond with the additional dependency information taught by Saboff, for the purpose of specifying the files required by a particular version of an assembly.

Conclusion

14.   The prior art made of record and not relied upon is considered pertinent to applicant's disclosure. U.S. Pat. No. 6,658,659 to Hiller et al. discloses a system for loading software modules of a particular version.

15.   Any inquiry concerning this communication or earlier communications from the

examiner should be directed to Michael J. Yigdall whose telephone number is (703) 305-0352.

The examiner can normally be reached on Monday through Friday from 7:30am to 4:00pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's

supervisor, Tuan Q. Dam can be reached on (703) 305-4552. The fax phone number for the

organization where this application or proceeding is assigned is 703-872-9306.

Information regarding the status of an application may be obtained from the Patent

Application Information Retrieval (PAIR) system. Status information for published applications

may be obtained from either Private PAIR or Public PAIR. Status information for unpublished

applications is available through Private PAIR only. For more information about the PAIR

system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR

system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free).

                                    MY   Michael J. Yigdall
                                              Examiner
                                              Art Unit 2122

mjy
April 26, 2004

                                              TUAN DAM
                                        SUPERVISORY PATENT EXAMINER